BAAN V

User Interface

User Manual U7034B US

Research and Development

Document information

Document

Number : U7034B US
Group : User documentation
Name : User Interface
Edition : B
Date : June 1997

Table of contents

About this document

This document describes the standard user interface (UI) for BAAN V.

The BAAN V UI uses the Object Action Model to accomplish a complete Windows look and feel. This model is the standard for the BAAN V UI.

Purpose of the Document
The purpose of this document is to inform the application developer about the backgrounds, the principles, the standards and the use of the BAAN V UI, so that applications can be developed efficiently and effectively.

Scope of the Document
This document works in close relationship with:

- the Data Access Layer for BAAN V
- the Coding & Text Standards
- the Programming Standards
- the Documentation Standards
- the BAAN V User Interface Reference Manual

References
This document is based on the following publications:

- The Windows Interface Guidelines for Software Design, Microsoft Press, 1995.
- Functional Design of the User Interface for BAAN Bravo
- BAAN IV GUI - Developer's Guide
- BAAN IV GUI - User's Guide
- Standard User Interface (TRITON Tools 5.0)

Overview
Chapter 1 addresses the design and interaction principles a UI must meet.

Chapter 2 addresses the UI components a developer can use to implement the interaction principles discussed in the previous chapter.

Chapters 3 address the standards for developing sessions and chapter 4 and describes methods to start sessions, especially zoom facilities and synchronization.

Chapter 5 addresses the standards for developing forms, chapter 6 gives information on working with form fields and chapter 7 on working with form commands.

Chapter 8 provides a checklist for the BAAN V UI.

1 User interface principles

1.1 Design principles

A well-designed UI is based on a development process that is centered around users and their activities. The user perspective is an important aspect of application development in general and this chapter addresses vital UI features for the user.

Application command
Users must have the impression that they are in control of an application, not that they are controlled by the application. For the UI, this consideration has the following implications:

In the relationship between a user and a computer, the user carries out the actions and plays an active part whereas the computer or the application is passive. In other words, the application can automate many actions, but it is up to the user to select and initiate the appropriate action. For example, the cursor must not jump to the next form automatically; this type of action must be initiated by the user.

Not all users have the same skills and expertise, so that is why the application actions and options must be shown in an intelligible way.

Finally, the application must be as interactive as possible. The user must be allowed to intervene, for example, by interrupting a process.

Directness
Users must be able to see the effects of their actions. Availability of information and options make users' jobs easier.

Consistency
Consistency in the application ensures that existing expertise can easily be applied in new tasks. Consequently, users learn faster and can devote more time to their work because they do not have to spend effort on learning new UI functions. It is essential therefore that the UI is standard throughout the entire application. This standardization allows users to familiarize themselves with the system as well as making the interface predictable.

Forgiveness
Users like to explore the application by trial and error. A good UI anticipates this attitude and provides a limited set of options: displaying a warning in case of (potentially) dangerous situations that might result in a loss of data integrity, or allowing actions to be undone.

Feedback
The application must always confirm what is performed after a user has initiated an action. There is nothing more frustrating for a user than looking at a screen that apparently does not show any reaction. The average user will not wait for more than a few seconds for a UI to respond. This confirmation will also prevent repeated input of commands.

Aesthetics
The design of what the user sees is an important aspect of application development. Visible attributes provide valuable information. On the other hand, each attribute on screen is liable to divert the user's attention. It is up to the designer to create a pleasant working environment that clearly contributes to the user's perception of the information without being distracting

This document presents important rules and guidelines for the design of the application's UI.

Simplicity
A UI must be easy to learn and use and it must provide access to all functionality of the application. Expanding functionality while maintaining the UI's simplicity is difficult to accomplish. An effective design considers both aspects. A few recommendations to keep the UI simple are:

- Limit the information on the screen to what is necessary for proper communication. For example, use short, clear-cut texts as action prompts.
- A clear layout and presentation contributes to the ease of use.
- Limit the number of menu items and required actions per screen.

1.2 Object action model

In a graphical UI environment, the object action model is the standard. First, the user selects an object and then the action to be performed on that object.

In the Baan environment, an object is a record. The record is selected in a multi-occurrence main window.

Then an action, such as update, insert or delete is selected. Depending on the action, a secondary window is usually opened that provides detailed information on the record and allows the user to view or change it, depending on the user authorizations.

2 User interface components

2.1 Introduction

In BAAN V there are six types of windows:

1 List Window
2 Synchronized dialog
3 Dialog
4 No window
5 Modeless window with menu bar/toolbar
6 Modal window with menu bar/toolbar

List window
All multi-occurrence sessions should be of this type. When this type is used, it can be used for both multi-occurrence sessions (with menu bar and toolbar) and for zoom sessions (without menu bar and toolbar). One definition is sufficient.

It depends now on the start mode, which you get:

- A modeless multi-occurrence (display or edit) with menu bar and toolbar
- A modal multi-occurrence display with the buttons, but without menu bar and toolbar.

The user interface components of the list window are detailed in the next paragraph.

Synchronized dialog and Dialog window
These types are used for the synchronized single-occurrence forms and for all type 4 forms. These forms will have no menu or toolbar. You can specify buttons on these forms. They will be positioned on the right hand side of the form.

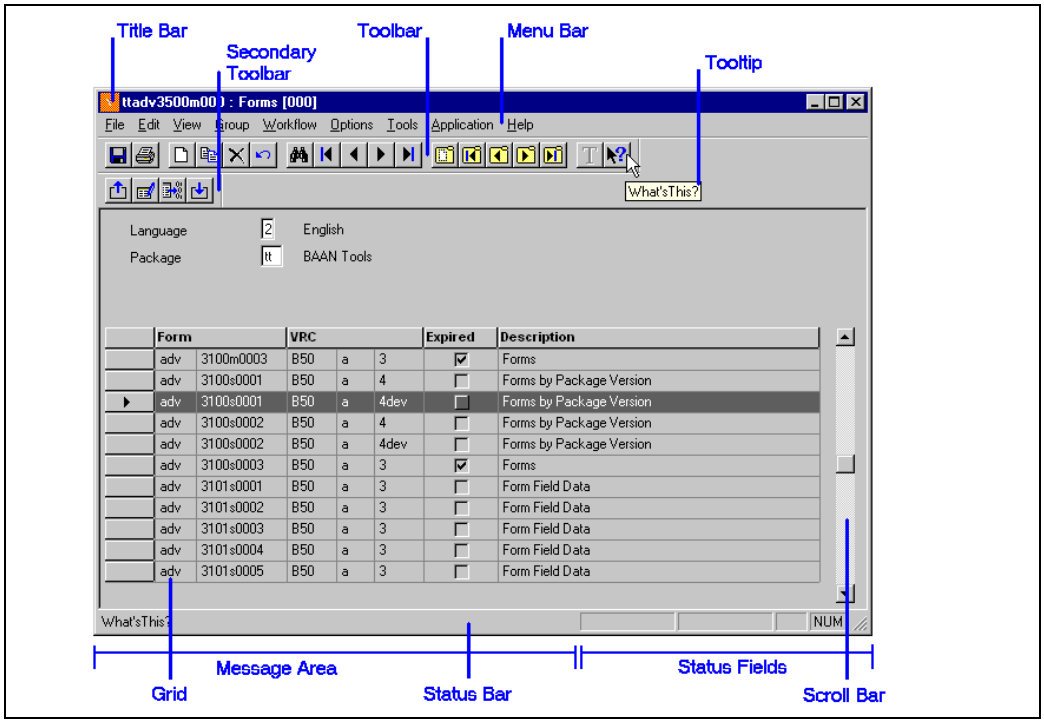
For describing their user interface components, both window types will be grouped together in the paragraph below on dialog window.

No window
These forms are started in the same main window as their parents.

Modal and Modeless window with menubar / toolbar
These window types are for backward compatibility only. They should not be used in new sessions and are not discussed separately here.

2.2 List window

Apart from the form definition displayed in the window, each main window has a title bar, a menu bar, a toolbar, a status bar and optionally a secondary toolbar. These components will be discussed in the following paragraphs.



2.2.1 Title bar

The title bar contains the name and code of the BAAN application, the company number of the BAAN user and the default windows controls.

If the title bar must show the session code, this feature can be activated using the User Data session.

2.2.2 Menu bar

Commands are distributed over the menu groups. There are three ways to activate a command from the menu bar:

- **Access keys:** Press Alt+underlined character (mnemonic) to open a menu group, followed by the underlined character of the item in the menu group.
- **Mouse:** Click on a menu group and hold down the mouse button. Drag the mouse to a menu item and release the button.
- **Shortcut:** In many menu groups, commands have a shortcut key. Press this key to start the corresponding command. For example, Ctrl+E is a shortcut key for Modify.

The following menu groups are session specific:

- **View:** Used to change the sort order of the records. The current order is marked with a check mark.
- **Application:** Contains all form commands.

2.2.3 Toolbar

The toolbar provides another way of starting the most frequently used commands. If the mouse pointer moves over a button, a short explanation is displayed in a ToolTip.

2.2.4 Grid

In BAAN V the multi-occurrence window is replaced by the grid. The grid provides the end user with much broader functionality. For example, the user can specify how wide the columns in the grid should be. It is also possible to split the screen.

2.2.5 Scroll bar

Multi-occurrence forms usually have a scroll bar for scrolling through the table. A scroll bar offers the following facilities:

To scroll	Do this
One record up/down	Click the up/down scroll arrow
One screen up/down	Click the scroll bar above or below the scroll box
To start/end of table	Drag the scroll box to the top or bottom of the scroll bar

BAAN is a multi-user, large scale database environment, so relative positioning of the scroll box could cause performance problems. Hence, the scroll box is displayed at three positions only: top, halfway, and bottom.

2.2.6 Status bar

In addition to a toolbar, the BAAN windows contain a status bar to show information that does not require a reaction such as a message concerning a successfully completed action.

The status bar contains a message area and four status fields.

Message Area

This area shows information (whenever available) and messages that need not be confirmed. Messages requiring confirmation are displayed in a dialog box.

Messages can be triggered by the program script, using the command:

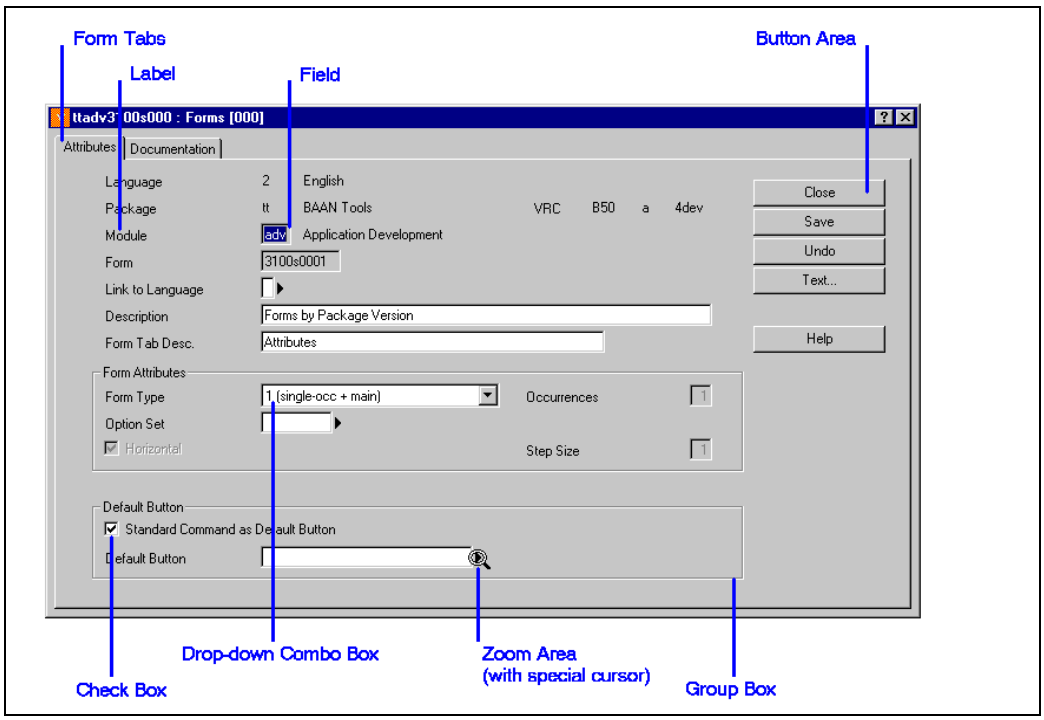
```
mess(string messcode(14), 0)
```

Status Field

The status fields display the status of the process like Modify or Add and the state of the keyboard (the Caps Lock and Num Lock keys).

2.3 Dialog window

Dialogs have a title bar in common with list windows. Specific components of this window type are form tabs and form buttons.



2.3.1 Button area

At the right hand side of single-occurrence form windows, an extra button area is available for standard and form commands.

2.3.2 Form tab

When there are multiple forms linked to a session, their presence is indicated by tabs shown in the window. These tabs should have labels that clearly indicate the general contents of their respective forms.

2.3.3 Label

In a GUI environment, labels and associated input fields are left aligned. Labels referring to domains are aligned according to the domain.

2.3.4 Field

Fields can be displayed as:

- input field
- display field
- disabled field
- read-only field

The last two modes can be programmed using a set of functions. See chapter 6 for more details.

2.3.5 Drop-down combo box

This box allows you to select an enumerated value. The key combination for opening the drop-down combo box is Alt-↓. Closing the box without changing the original value is performed by clicking outside the box or by pressing <Esc>.

2.3.6 Check box

These small boxes represent a Boolean type field: true/false, yes/no, 1/0, on/off, and so on. The value can be changed with the mouse, or by pressing the spacebar (key combination). The relevant domains can be set using the 'Maintain Domains to be Displayed as Check Box' session.

A check box is used to support options that are either on or off, and is displayed as a square box with an accompanying label. When the choice is set, a check mark appears in the box. Check boxes are typically used for independent or nonexclusive choices.

The check box should be displayed to conform to the Windows Interface Guidelines. Here, you will be reminded that the label should be placed on the right side of the box directly behind it. The label currently displayed on the left should be removed.

The check box can be checked or unchecked by clicking on the label.

2.3.7 Group box

A group box is used to cluster functionally related information. The title of the group box describes a collective feature of the clustered elements.

2.3.8 Zoom area

These small triangles allow you to zoom in on the field. The cursor changes into a magnifying glass (as is displayed in the figure above) if it appears above this area.

2.4 Progress indicator

This window shows the user that the system is processing data in the background. A response time of more than a few seconds is not acceptable, which is why feedback is provided in the form of a process indicator.

The essence of this busy window is not to show the expected duration of the process but merely that the system is doing something.

Process indicator control is triggered by the application developer. The progress indicator can contain user-defined labels. The percentage bar and corresponding label are optional.

The process indicator replaces the BAAN IV processing forms. Fields showing the process' progress, or special progress forms, are no longer allowed.

More details are available in the Data Access Layer documentation.

2.5 Message box

If you usie the mess(messcode, 1) command, a message dialog with a message specified in the data dictionary can be displayed. In addition the icon that belongs to the message is also displayed as an Information message, a Warning, or a Critical message icon.

Information	Provides information about the results of a command. Offers no user choices.
Warning	Alerts the user to a condition or situation that requires the user's decision and input before proceeding, such as an impending action with potentially destructive or irreversible consequences.
Critical	Informs the user of a serious problem that requires intervention or correction before work can continue (avoid this type as much as possible).

It is possible to define help text that will be displayed when the user chooses the Help command button within the message box.

The user assistance provided by a Help command button differs from the typical 'What's This?' form of help. Command button help should provide an overview, summary assistance, or explanatory information about the displayed message. For example, this help can provide more information about possible causes and remedies for the reason for which the message was displayed.

Use the message box in situations in which it will relay useful additional information. Make use of the message Help much as possible in order to help the user continue his work. Consider the Message Box Help as additional assistance and not as a successor to the 'What's This?' kind of help. Try to understand the situation of an inexperienced user when writing the help text.

3 Sessions

3.1 General

Technically, a session is the combination of a main table, a program script, a form, and reports. Session codes and names are essential, because they constitute the framework of the application software and corresponding documentation.

In BAAN V, five types of sessions are distinguished:

- Multi-occurrence sessions
- Single-occurrence sessions
- Processing sessions
- Print sessions
- Graphical sessions

For each of these types, the coding, naming and start option are addressed.

3.1.1 Coding

The basic code layout for a session is: ppmmmsfnnp000

- pp = package code
- mmm = module code
- s = submodule code
- f = function code
- nn = sequence number
- p = process type

The values of the function code and the process type depend on the session type and are discussed accordingly.

A process type can be:

- m = multi-occurrence
- s = single-occurrence

3.2 Multi-occurrence sessions

In a multi-occurrence session, multiple records are shown simultaneously in a grid.

3.2.1 Usage

Multi-occurrence sessions can be classified according to usage:

- For selecting in the object action model. Such sessions are synchronized with a single-occurrence edit session. This is the standard for BAAN V.
- For editing. Such sessions cannot be synchronized with single-occurrence sessions.
- For viewing.

Synchronized with single-occurrence session

In the object action model, the multi-occurrence session is used to select the object from the data in the main table and subsequently carry out an action.

This is the preferred way to use multi-occurrence sessions.

Multi-occurrence with edit facility

Because the most important criterion for a Windows application is its usability, in some situations a multi-occurrence session with edit facility is allowed.

The only condition for a multi-occurrence edit session is that all columns of the grid must fit on an 80-position form. Because all fields fit on the multi-occurrence form, a single-occurrence display of the data is not necessary. If the fields do not fit on an 80-position form, the solution is to synchronize a multi-occurrence display session and a single-occurrence edit session.

Examples for such sessions are those in which master tables with only a code and a description are maintained.

Multi-occurrence display

In the BAAN applications, multi-occurrence sessions are also used to display, for example, the costs history of an item or a project budget.

Such display sessions by definition cannot be synchronized with another session unless programmed explicitly by an application developer.

3.2.2 Coding

The table shows the function codes and process types for each usage category:

Usage	f	p	Code	Synchronized with
Selection	5	m	ppmmms5nnm000	ppmmms1nns000
Edit	1	m	ppmmms1nnm000	not
Overview	5	m	ppmmms5nnm000	not

3.2.3 Programmer Information

Synchronized with single-occurrence session

- Session Name
  Because of the object action model and the new authorization facilities, the difference between maintenance and display sessions no longer exists for the end user. These factors are why these words are no longer allowed in session descriptions.

  Consequently, the description of a multi-occurrence session is identical to the table it acts upon.

- Session Form
  A multi-occurrence session that is synchronized with a single-occurrence session contains a form of up to 132 positions. The form contains the most important fields of the record so that the user can select the proper record.

  One criterion to include fields in a form is that they are part of the primary or the alternate key of the table, which enables the user to apply different sorting criteria.

- Synchronization link
  To indicate to the standard program that the multi-occurrence session must be synchronized with a single-occurrence session (in the before.program section of the multi-occurrence session) use the command:

– set.synchronized.dialog(session code of SO session)

Multi-occurrence with edit facility

- Session Name
  Because a UI must be consistent, the word maintain may not be part of the session description.

  Consequently, the description of a multi-occurrence session with an edit facility is identical to the table upon which it acts.
- Session Form
  As stated earlier, a multi-occurrence session with edit facility can have no more than a single, 80-position form containing all fields.

Multi-occurrence display

- Session Name
  Because a UI must have a consistent presentation, the word display cannot be part of the session description. For historic data, the session description will become for example Sales Order History. For other listings, the session description will be for example Cost Price Summary.
- Session Form
  A multi-occurrence display session can contain multiple forms of up to 132 positions. In essence, the session consists of a series of sheets with a grid. The grids are only filled with data when the sheet is active, resulting in better performance.

3.3 Single-occurrence sessions

In a single-occurrence session, the data of a single record is shown.

3.3.1 Usage

A single-occurrence session can be used to:

- Edit (modify, insert) a record. This is always done in synchronization with a multi-occurrence session.
- Display a record. This is always done in synchronization with a multi-occurrence session.
- Edit parameters.

The single-occurrence edit session is the standard for BAAN V.

Single-occurrence with edit facility

This type of single-occurrence session is mainly used in synchronization with a multi-occurrence display session.

If a record has more fields than fit on an 80-positions multi-occurrence form, editing and displaying of all record data is done in a single-occurrence session.

Single-occurrence display

Once in history, data can no longer be maintained. If a multi-occurrence display session is not feasible, the object action model can be implemented, featuring a single-occurrence display session.

Single-occurrence parameter setting

Each module has only one set of parameters. This configuration makes it illogical to access parameters through the object action model. It must be possible to start the single-occurrence parameter sessions directly from the menu browser.

3.3.2 Coding

The following table shows the function codes and process types per single-occurrence session usage:

Usage	f	p	Code	Synchronized with
Edit	1	s	ppmmms1nns000	ppmmms5nnm000
Display	5	s	ppmmms5nns000	ppmmms5nnm000
Parameters	1	s	ppmmms1nns000	not

3.3.3 Programmer information

The following information on the session description and the session form applies to all single-occurrence sessions.

- Session Description
  The difference between maintenance and display sessions no longer exists, so these words are no longer allowed in the session description.

  Consequently, the description of a single-occurrence session is identical to its associated synchronizing multi-occurrence session.
- Session Form
  A single-occurrence session can contain multiple 80-position forms. Forms are functionally identified by a description. Consequently, fields are grouped on forms according to their function.

3.4 Processing sessions

Processing sessions submit data from multiple tables to operations and subsequently store it as new records in other tables.

Processing sessions are also used to print items such as error reports or invoices.

3.4.1 General

Sessions like printing invoices and the associated financial transactions are not print, but a processing sessions. The session description must reflect this behavior.

3.4.2 Coding

The table shows the function codes and process types for processing sessions:

Usage	f	p	Code	Synchronized with
Processing	2	m	ppmmms2nnm000	not

A processing session usually processes multiple records, which is illustrated by its process type: *m*.

3.4.3 Programmer's information

- Session description
  The session description describes the action taking place. For example, the description for an invoicing session would be Invoice Sales Orders.
- Session Form
  The format of the form for a processing session is free, and multiple forms are possible.

  The first form at minimum contains the selection ranges for the data to be processed. Other data that is relevant for the process such as commands that can be toggled using check boxes can be placed on a second or later form.

  A process is started by pressing the Continue button. The label on the Continue button must show the process' action. A Cancel button must be available as well.
- Progress Indicator
  A progress indicator informs the end user about the progress of a process.

3.5 Print sessions

Print sessions are used to print data in reports, which are sent to an output device.

3.5.1 General

In a print session no database data is changed.

3.5.2 Coding

The following table shows the function codes and process types for print sessions:

Usage	f	p	Code	Synchronized with
Print	4	m	ppmmms4nnm000	not

A print session usually processes multiple records, which is illustrated by process type 'm'.

3.5.3 Programmer's information

- Session description
  The session description must show that it is a print session.
- Session Form
  The format of the form for a processing session is free, and multiple forms are possible.

  The first form, at minimum, contains the selection ranges for the data to be printed. Other data that is relevant for the process such as commands that can be toggled using check boxes can be placed on a second or later form.

  It must be possible to start the process by pressing the Continue button. The label on the Continue button is OK. A Cancel button must be available as well.
- Progress Indicator
  A progress indicator informs the end user about the progress of a process. Since performance is an issue here, it is not recommended to use the total number of records to be processed. The most important consideration is that the user gets feedback.

3.6 Graphical sessions

In a graphical session, data is displayed in non-character format.

3.6.1 General

Graphical display may be available for:

- A graph (chart) for capacity availability
- An object browser for tree structures
- A network structure for networks
- A planning board

3.6.2 Coding

The table shows the function codes and process types for graphical sessions:

Usage	f	p	Code	Synchronized with
Graphical	7	m	ppmmms7nnm000	not

In a graphical session, data is usually shown for multiple records, which is illustrated by process type *m*.

4 Programming sessions

4.1 Introduction

In this chapter, several mechanisms are described that start one session from another:

- Zooming from an input field
- Synchronizing sessions
- Starting a session from a 4GL script

4.2 Zooming from fields

The user can zoom in on an input field to retrieve a record (for example, the unit corresponding to an item). In that case, a modal multi-occurrence display-only window is opened, the selected record is exported to the main window by double clicking and the zoom window is closed.

The zoom window can contain the following buttons:

Buttons on Zoom Window	
Button	**Action**
OK	Export the selected record and close the zoom window. This should be the default button.
CANCEL	Close the zoom window without exporting the record to the calling session.
FIND	Search for a specific record.
SORT BY	Change the sort order of the records.
QUERY	Display the records matching a user-defined query.
DISPLAY	Open a modal secondary zoom window. This window is a single-occurrence display-only window that shows details of the record. An OK button is available to close the window.
NEW	Open a modal secondary zoom window. In this single-occurrence edit window a new record can be inserted.
PRINT	Open a print dialog.
ROTATE CURRENCY	Change the currency format.
HELP	Display the session.

The main zoom window and the secondary zoom window are modal types, so the window must be closed before the user can continue in another window.

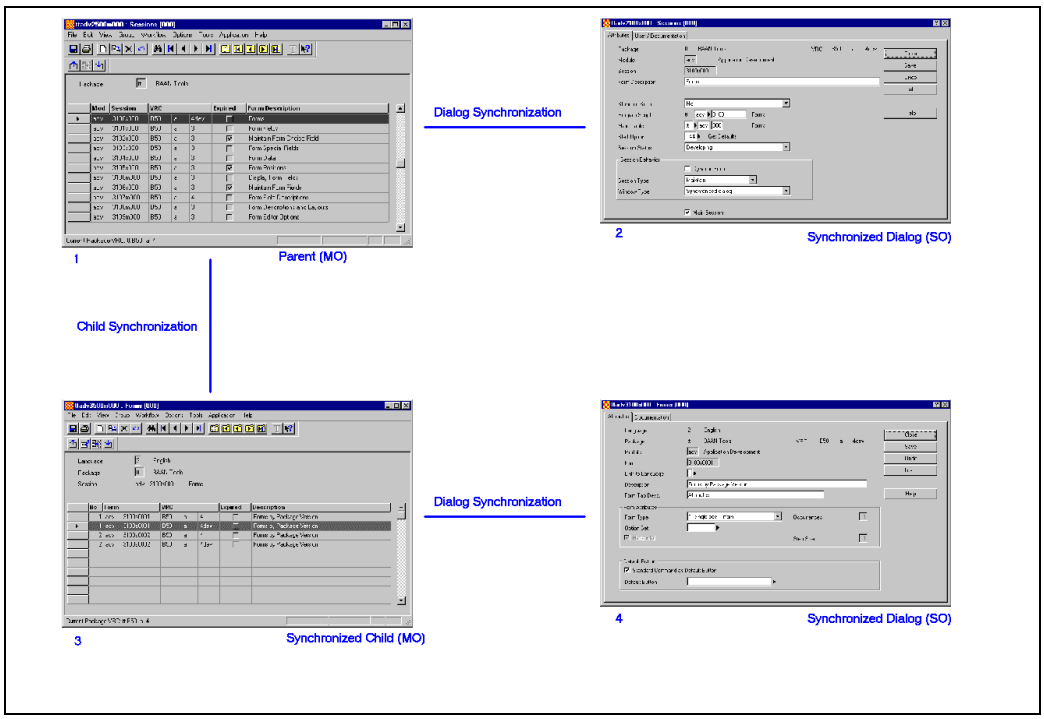
4.3 Synchronizing sessions

In the object action model, several windows are shown simultaneously. These windows can be synchronized. In other words, actions performed on one of the windows will have a direct effect on the other window.

There are two independent synchronization models implemented in Baan V:

1. Child Synchronization. This model is intended for synchronizing two processes with different main tables. The child (controlled session) is started with the function start.synchronized.child, and synchronization is only done when calling the function synchronize.with.child.
2. Dialog Synchronization. This model is handled by the standard program. The time of synchronization cannot be influenced in the User Interface script. This makes it easier to program but also less flexible than child synchronization.

An example with both types of synchronization is displayed below:



4.3.1 Dialog synchronization

Overview

Dialog Synchronization synchronizes two processes with the same main table. The synchronization occurs between the multi-occurrence parent session and a single-occurrence dialog.

Every time an occurrence in the parent is double clicked, dialog is updated with information from the selected record. For example, the multi-occurrence form displaying a list of items and the single-occurrence form displaying a single item. The multi-occurrence session can also be used as a zoom dialog. In this case, the session will appear as a display-only modal dialog with an OK and a Cancel button.

Parent session (Multi-occurrence)

The main window is a multi-occurrence, display-only window. The records can be browsed using the scroll bar or the <Page Up> and <Page Down> keys. A record is selected by clicking it with the mouse.

Once a record has been selected, the standard program automatically opens the secondary window if one of the following actions is initiated:

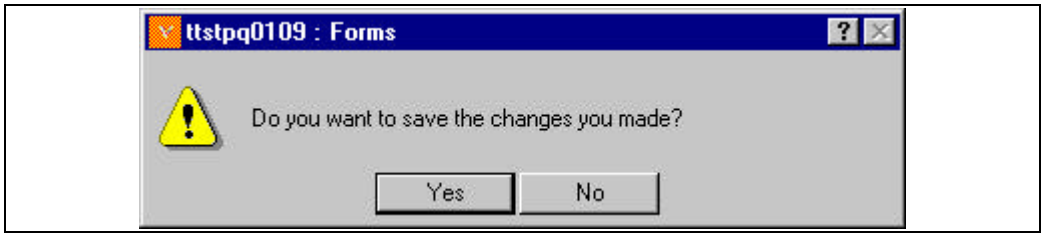
- Insert record
- Edit record
- Duplicate record

Double-clicking a record opens the secondary window with the standard program in edit or display mode, depending on the authorizations.

A display-only action can be selected from the menu bar, which automatically opens a display-only secondary window.

The remaining actions in the main window do not call a secondary window.

Secondary windows, if any, are closed along with the main window. If a record has not been saved, the user is prompted to save it now.



Synchronized dialog session (Single-occurrence)

Depending on user authorization, the secondary window is either a single-occurrence display or a single-occurrence maintenance window. This window can contain multiple forms.

A secondary windows features no menu bar or toolbar. In the title bar the common Windows 95 buttons for Close and Field Help are available.

The form can contain buttons to start the following actions:

- Save record
- Form commands
- Session help
- Query by form
- Undo

Because the secondary window is modeless, the main window remains active while the user is working in the secondary window, which means that the following actions are still possible in the main window:

- Selecting another record
- Deleting a record
- Inserting a record

The standard program makes sure that, after a record has been saved, the data in the secondary window matches the data in the main window.

Implementation

Dialog synchronization can be implemented as follows:

1. Create two separate sessions. One contains the multi-occurrence form with menu bar and tool bar and a window type of list window. The other session contains the single-occurrence dialog of type synchronized dialog.
2. Add to the before.program section of the multi-occurrence session the following code:

```
before.program:
    set.synchronized.dialog("ses.code")
```

   Where ses.code is the session code of the synchronized single-occurrence session.

New variable synchronized.reason

In the synchronized dialog a new variable is set:

```
synchronized.reason
```

This variable is read-only and indicates the command that caused the synchronization event:

- ADD.SET
- DUPL.OCCUR
- MODIFY.SET
- DISPLAY.SET

This variable is available in all sections in a synchronized dialog. Note that this variable is not set in synchronized children because the synchronization there is caused 4GL code instead of a form command.

Using view fields

Synchronization between a multi-occurrence session with view fields (Form Type 3) and a single-occurrence session (Form Type 1) requires special treatment:

1. Define the Form Type of the single-occurrence session as type 3 (multi-occ+view+main) with one occurrence.
2. Leave the Form Type of the multi-occurrence session as type 3.
3. The view fields should be made of Field Type display and the Multi-Occurrence flag should be No.
4. The Multi-Occurrence flag of the occurrence fields should be Yes.

To summarize:

Form types	
Parent form	**Dialog form**
type 2	type 1
type 3	type 3 with 1 occurrence

Session settings

To implement a successful synchronization, the settings of the session data should be as follows:

Session Settings		
	Parent session	**Dialog session**
Session Type	Display	Maintain
Start Option	44 (get defaults)	0 (no start option)
Window Type	List Window	Synchronized Dialog
Main Session [1]	Yes	No

[1] The session setting Main Session means that it is possible to start the session from the menu browser.

4.3.2 Child synchronization

Overview

Child synchronization can be used between sessions that operate on different main tables. The primary key of the parent's main table must be a subset of the primary key of the child's main table.

For example, the parent session uses orders (where the primary key is the order number) and the child session uses order lines (where the primary key is the combination of the order number and the order line number). The order number in the order lines table is a foreign key referencing the orders table.

Implementation

Child synchronization can easily be implemented by calling the following function:

```
function long start.synchronized.child(const string session_code [,
            string parent.var, string child.var] ...)
```

The first argument is the session code to be started as child. Other arguments come in pairs: the first of these two is the variable name of the parent main table, and the second is the equivalent variable in the child table.

For every synchronization event, the child variables get their values from the parent variables and a refresh is done.

```
synchronize.with.child(user.x.child.id)
```

Note that if the user.x.child.id process does not exist, the child.x.id value is set to 0.

```
declaration:
    long    field.proc
    ...

before.program:
    field.proc = 0
    ...

function extern field.process()       | called by Form Command
{
    if field.proc then
        synchronize.with.child(field.proc)
    endif
    | Note : field.proc is a reference variable that is set to zero when
    | child cannot be found.
    if not field.proc then
        field.proc = start.synchronized.child ("ttadv4127m000",
            "ttadv420.vers",          "ttadv422.vers",
            "ttadv420.rele",          "ttadv422.rele",
            "ttadv420.cust",          "ttadv422.cust",
            "ttadv420.cpac",          "ttadv422.cpac",
            "ttadv420.cmod",          "ttadv422.cmod",
            "ttadv420.flno",          "ttadv422.flno")
    endif
}
```

4.3.3 The synchronization process step-by-step

In the picture on page 4.2 four windows are shown: windows 1 and 3 are main windows (multi-occurrence parent sessions) and windows 2 and 4 are secondary windows (single-occurrence dialog sessions).

Window 1 and 2 form a set operating on a certain main table, and window 3 and 4 form a set operating on another one.

The windows of each set are synchronized using dialog synchronization. The sets themselves are synchronized using child synchronization.

Three synchronization processes will be described step-by-step:

- To display all four windows:
  - Start up the window 1 for instance from the menu.
  - Open window 2 for instance by double-clicking a record in window 1.
  - Select a record in window 1.
  - Using a form command on window 1, start window 3.
  - Open window 4, for instance by double-clicking a record in window 3.
- Select another record in window 1 and by double clicking perform the default action. The consequence is that in window 2 the data of that record is displayed.
- Again, select another record in window 1 and activate the form command to start window 3. The consequences of this action are:
  - The data in window 2 are not refreshed by the detail information of the selected record.
  - In window 3 the children of the selected record are shown.

4.4 Starting sessions in 4GL

With the function start.session, you can start other sessions from a program script. This function can be called from anywhere in the script and is not restricted to any section. This function is the replacement for zoom.to$, start.main.session, and other functions for starting sessions.

The syntax of this function is:

```
string start.session(long mode, const string session.code, const string zoomname,
                     const string returnfld)
```

The arguments have the following meaning:

Argument	Description
mode	Can be either MODAL (blocking the parent, until child is exit) or MODELESS (two concurrent sessions).
session.code	The code of the session to be started.
zoomname	The name indicating the calling process. This name is used in the zoom.from.... sections in the called process. Beware that if this argument is not filled, no zoom from sections are executed, not even the zoom.from.all section.
returnfld	The name indicating the variable that is returned by the function. The contents of the variable are copied in the return value of start.session This argument is only applicable if the mode is MODAL. If an empty string is given, the exit value of the zoom process is returned. Note that this argument should be the name of a variable, not the variable itself.

5 Forms

5.1 Introduction

The user accesses a session through the corresponding window. The form is the part of the window that provides the database access, so forms must be developed from the user perspective.

In BAAN V, three types of forms are distinguished:

1. Single-occurrence forms
2. Multi-occurrence forms
3. Free-format forms

The common aspects of the form types are addressed first, followed by the characteristics per individual type.

Similarities are:

- The codes of the forms
- The names of the forms
- The guidelines for form development
- Form commands

The differences occur in:

- Commands
- Buttons
- Form layout

This chapter will be concluded with a paragraph on a new grid feature: totals in grid.

5.2 General

This section addresses the common aspects of the form types: the coding and naming of forms and the guidelines for form development.

5.2.1 Coding

Form coding format: ppmmmsfnnp000c

- pp = package code
- mmm = module code
- s = submodule code
- f = function code
- nn = sequence number
- p = process type
- c = form sequence number

In this coding format c can be any number from 1 to 9 or any letter from a to z.

The first 13 characters of this code usually match the code of the corresponding session.

5.2.2 Form descriptions

In a window, the form description is not displayed. The session description is shown in the title bar.

Forms are used to group data for the user, so form descriptions must be clear.

The following rules apply:

- If there is only one type of data covered by the form, a description is considered redundant. Selection sessions are a good example. However, it is advisable to enter a form description anyway.
- In multiple forms:
  - The data is distributed over the forms by function.
  - Each form is given a description that covers the data contained in it.

This setup avoids descriptions such as General 1 or Miscellaneous 2.

5.2.3 General programmer's information

This section addresses the general guidelines. First, for the development of forms, then for the input and display sequence, for labels and field alignment and, finally, for group boxes.

Rules for form development

The general rules governing the development of forms are:

- All forms, with the exception of graphical forms, are created from standard form types. Additional rules are provided in the chapters below.
- Accentuating sections of the form by means of labels in reverse or underlined mode is not allowed.
- All forms for a session have the same size and box attributes.
- Key fields are in the same position across forms.
- Fields representing an On/Off, Yes/No, 1/0, True/False, or any Boolean value, are implemented using a check box. For check boxes the following applies:
  - The field (check box) is positioned in front of the label and left aligned in the column.
  - The check box and its label are not separated by spaces.
  - The check box must be displayed as two characters.
  - Check boxes are grouped, if possible.
- A label is separated from the associated field by a single space. Colons are not used.
- In case two From-To range fields are placed on the same row, there should be five spaces between those fields. In case more range fields are placed beneath each other in two columns, there should be five spaces between the largest From field and the To field. The other pairs of range fields are left aligned with these two.
- The box type for forms is Normal. BAAN V makes no distinction between main and sub sessions.
- The first label is positioned at row 4, column 4.
- The outer size of a form is 22 rows by 80 columns. The inner size, the space actually available for designing, is 18 rows by 78 columns.

Rules for sequencing of fields

The standard input sequence is from top to bottom and from left to right.

- Display fields showing a description are positioned to the right of the associated field and displayed upon entry in that field. If the description cannot be positioned as described above, it is positioned under the associated field and both fields are left aligned.
- The from/to range fields are placed in pairs next to each other. When there is insufficient room to do that the fields are positioned beneath each other. In the latter case, all from fields are placed in one column and all to fields in another.
- If a field can only be entered dependent on the value of another field, this situation is considered an input hierarchy. The hierarchy is indicated by putting dependent fields below their determining fields. Enabling or disabling the dependent fields shows the user whether the fields are accessible or not.
- If a field has mandatory input that must be available elsewhere in the system, a zoom command is available to a dialog with these values. For numeric values, such a zoom command is considered useless.
  - Make the zoom user-friendly, in order to prevent the user from making unnecessary mistakes. Use a query.extension to build the subset of zoom data.

Rules for Labels and Fields

The general rules governing the use of labels are:

- The recommended label field length is 20 positions. For column headers, a length of 15 positions is recommended.
- In the layouts of menus, forms, and reports, reserve the maximum number of free positions for label fields, taking into account the following:
  - Reserve at least the label text in the development language multiplied by the highest label length multiplication factor for the primary languages.
  - Reserve positions for top, bottom, left and right margins, as well as between fields.
- Additional rules for label length:
  - Frequently occurring, generic labels have fixed codes and lengths as defined in the Software Coding Standards.
  - Reserve the maximum number of positions for label fields.
  - The absolute minimum for a label field is three positions, because it is not always possible to devise an adequate abbreviation in all natural languages.
- The maximum height for a label field is three lines.
  - If multi-occurrence layouts contain label fields of more than one line, all other label fields in the same row have the same number of lines reserved.
  - In such cases, one-line labels and two-line labels are always top- aligned.
  - Try to avoid three-line labels.
- Labels are left aligned. Labels in multi-occurrence forms are aligned according to the field domain.

Rules for using fields are:

- Fields in a form are left aligned. The only exceptions to this rule are interrelated numeric fields in a column: for example, unit prices and total price. Such fields are aligned at the decimal point.

Rules for group boxes on forms

- Fields that are functionally related are grouped in a group box. One or more group boxes constitute a form.
- If a form has only one group box, the group box as such is omitted because the form takes over the grouping function.
- Key fields are not underlined or grouped in a group box.
- A group box has a label in the top line, starting three positions from the left corner. The attributes of this label are:
  - Fill print effect is No
  - Alignment is Left
- Standard group box labels are:
  - Selection Range
    For print and processing sessions which require a selection
  - Print Commands
    For print sessions that require some command setting
  - Process Commands
    For processing sessions that require some command setting
- Between the vertical left group box line and the fields in the group box, one space must be reserved.
- Between the vertical right group box line and the labels and/or fields in the group box, a minimum of two spaces must be reserved.

5.3 Multi-occurrence forms

A multi-occurrence form is used to present the data of multiple records. In BAAN V, multi-occurrence forms are based on a grid.

5.3.1 Form layout

For an example of the layout of a single-occurrence form, see paragraph 2.2: List Window.

5.3.2 Programmer's information

Additional rules for the development of single-occurrence forms:

- A multi-occurrence form is up to 132 positions wide. The following exceptions are allowed:
  - History or summary sessions may feature multiple 132-column forms.
  - Multi-occurrence editing sessions feature a single, 80-column form.
- A multi-occurrence form can contain multiple view fields.
- A multi-occurrence form only contains horizontal occurrences.
- The initial number of occurrences is 13.

- The columns in a multi-occurrence form are separated by two positions.
- The length of the label field for a column is equal to or larger than the length of the column field.
- A blank row is inserted between the view fields and the label fields of the multi-occurrence section.
- A blank row is inserted between the label fields and the first record.

5.4 Single-occurrence forms

A single-occurrence form is used to present the data of a single record.

5.4.1 Form layout

For an example of the layout of a single-occurrence form, see paragraph 2.3: Dialog Window.

5.4.2 Programmer's information

Additional rules for the development of single-occurrence forms:

- A single-occurrence form can comprise multiple forms.
- A blank row is inserted between the last primary key field and the first record field.
- Units like kg are positioned between the label and the field in a separate, right-aligned field.
- A description in a form following the table field is separated from it by two spaces.
- Forms should be used instead of arranging fields in multiple columns. If columns cannot be avoided, they are separated by two spaces.

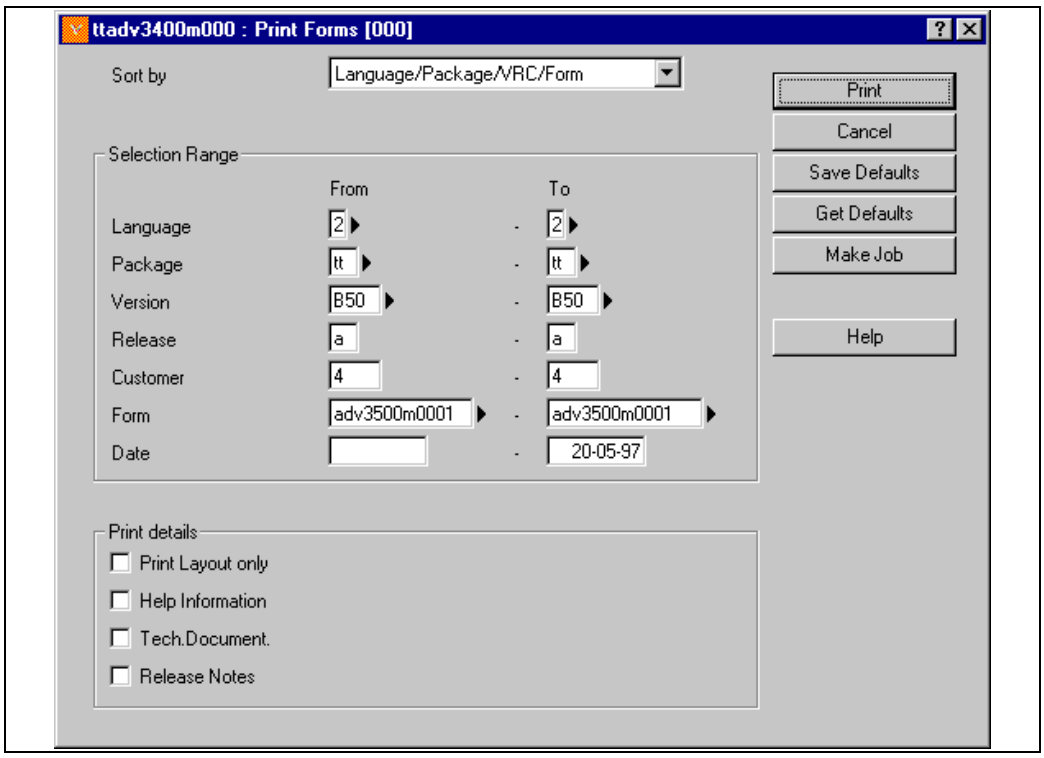
5.5 Free-format form

A free-format form is used to enter data for processing and print sessions. A free-format form resembles a dialog, it has no menu bar, toolbar or message/status bar. This form does however contain the following buttons:

- Continue (preferably with a more descriptive label like Print)
- Cancel
- Save Defaults
- Get Defaults
- Make Job
- Help

A free-format form is used to enter data for processing and print

5.5.1 Form layout



5.5.2 Programmer's information

Additional rules for the development of free-format forms:

- Multiple forms are possible.
- Group boxes are advised for the Selection Range and the Print Commands or Process Commands.
- Sort by functionality should be implemented by means of an enumerated.
- The progress box is replaced with the Progress Indicator.

6 Programming form fields

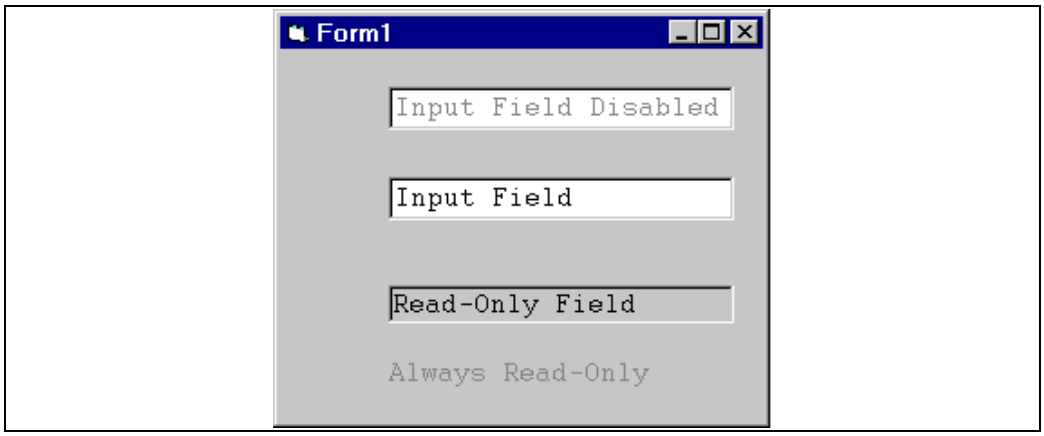
6.1 Introduction

This chapter introduces a number of enhancements to the handling of fields on forms.

6.2 Disabling fields

6.2.1 Overview

It is possible to enable and disable fields using functions in a 4GL script. When disabling a field it can be either disabled or read-only. The difference between these modes can be seen in the next picture:



6.2.2 Implementation

To use these functions the following statements have to be added to the script:

```
disable.fields([long mode,] <field> [, <field>] ...)
enable.fields(<field> [, <field>] ...)
```

Where mode is either disable (default) or read-only and `<field>` is a field name, followed by an occurrence number (only if field is a multi-occurrence field).

For example, when tccom010.beca and tccom010.dsca are single occurrence fields, these fields can be disabled as follows:

```
disable.fields("tccom010.beca", "tccom010.dsca")
```

To display tccom010.beca and tccom010.dsca as read-only fields in a grid the following code can be used:

```
disable.fields(READONLY, "tccom010.beca",actual.occ, "tccom010.dsca", actual.occ)
```

For disabling an array field tfmod100.perd(10) the code could look like:

```
disable.fields("tfmod100.perd(10)")
```

Note that for arrays no extra argument is added. You should use parentheses [()] in the field name. If no array element is specified, all elements are disabled.

Segmented fields can be disabled by adding the .segment.<id> string to the field name. For instance, if the first and second segment of the field tcibd001.dfit must be disabled, the following command should be used:

```
disable.fields("tcibd001.dfit.segment.1", "tcibd001.dfit.segment.2")
```

Note that if you want to disable all segments you leave out the segment names:

```
disable.fields("tcibd001.dfit")
```

Finally, UTC fields can be disabled by appending .date or .time to the field name. For instance, to manipulate the UTC field ttadv500.fidt:

```
ttaad500.fidt = 0
        display("ttaad500.fidt.date")
        display("ttaad500.fidt.time")
        disable.fields("ttaad500.fidt.date",
                       "ttaad500.fidt.time")
```

The functions can be used in the following sections:

before.display.object
After the data has been read from the database, fields can be initially enabled and disabled in this section before the fields are displayed on the form.

when.field.changes
This will work for browsing through records and for moving to the next field after input.

Note that the before.program section cannot be used.

These functions and sections make it possible to disable a field conditionally as soon as another field changes. This results in a good Windows look and feel. The field is now disabled immediately instead of at the moment it receives the focus (as was formerly the case with attr.input = false in the before.input event).

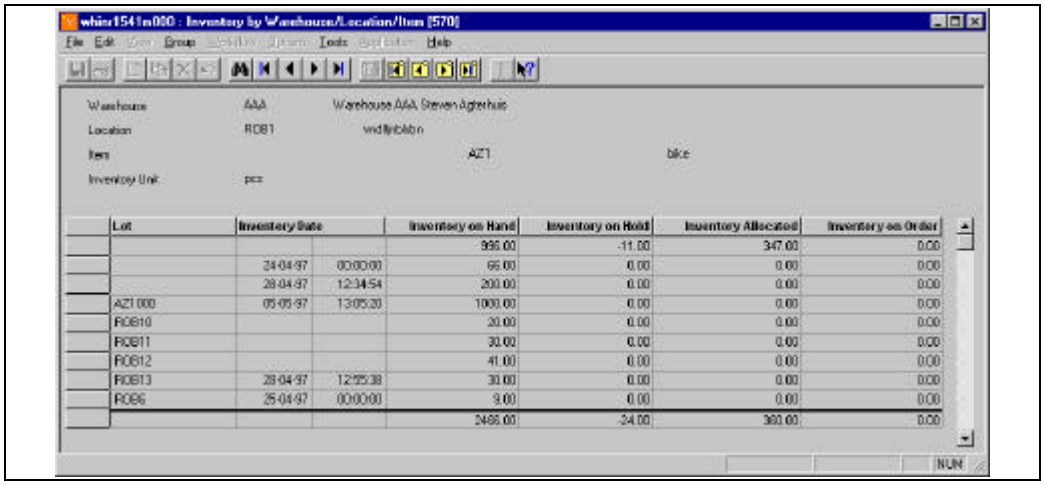
The attr.input variable has therefore become obsolete and should be replaced by using the disabling/enabling functions as follows:

- In new forms, use the enable/disable field functions.
- In old forms the attr.input must be replaced by disable.fields(fattr.currfld$). For the fields where this is done, also add enable.fields(fattr.currfld$) to the after.field section.
- A conversion tool can help with the old forms by automatically replacing the attr.input by the enable/disable function.

6.3 Total fields in a grid

6.3.1 Overview

In BAAN V, it is possible to add a total line to the grid.



6.3.2 Implementation

You only have to modify the UI script in order to add a total line to your grid:

1 Add the following code fragments in your UI script

```
before.program:
    fattr.total.line = true
```

2 For every update of the total fields add:

```
…
   display.total.fields(“field1”, value1, “field2”, value2)
   …
```

Where field1 and field2 are defined as multi-occurrence form fields on the form and declared in the UI script. Of course, this can also be a table field.

Restrictions

- Total have the same display properties as the field above.
- Array fields fields cannot be disabled or enabled (they are always read-only).
- The total fields cannot be displayed in the total line.

Example :

```
before.program:
    fattr.total.line = true
before.display.object:
    total1 = total1 +  whwmd000.amnt1
    total2 = total2 +  whwmd000.amnt2
    if actual.occ = filled.occ then
        display.total.fields(“whwmd000.amnt1”,total1,
                             “whwmd000.amnt2”,total2)
    endif
```

6.4 Segmented fields

6.4.1 Overview

Field Segmentation is a feature used for dividing a form field into multiple segments. This feature offers the possibility to structure the data within a single field. The definition of this segmentation is performed at the customer's site by means of a session. With this session, the user can specify which domains have to be displayed into different segments. In addition, different attributes can be assigned to the segments.

6.4.2 Implementation

After reading the form dump, all fields have to be checked for the presence of a segmented domain connected to it. If so, the field should be split into the segments, and for each segment an object has to be created with an optional zoom area. These objects have to be sorted in a separate array.

The standard program will only check the field (and carries out the field sections) if the focus goes to another field. A change of the focus within the fields will not lead to the field sections being carried out.

Known restrictions are:

- There should be enough space on the form to enlarge the field area.
- If a segment is set to numeric, only digits (0-9) are allowed.
- Field Specific Commands will not be activated for segmented fields. In other words, characters without carriage return cannot be tested in the on.input section of the script.
- Zooming on the segment level will not trigger the field zoom sections.

7 Programming form commands

7.1 Introduction

In BAAN V the ten user options, the zoom sessions and the cont.process command are merged into a new type: Form Command. The number of Form Commands for each form is (in theory) unlimited, but the user should be able to comprehend all these commands.

Other advantages of Form Commands are:

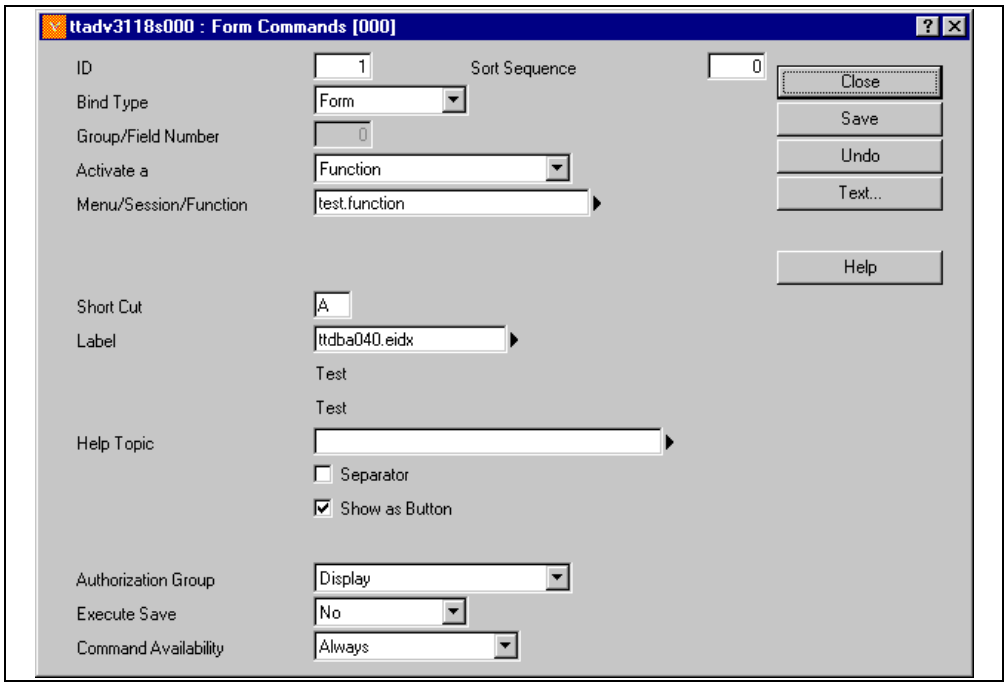
- Automatic enabling/disabling of the command, depending on number of selected records.
- Automatic removal of the command when the user has no authorization.
- The commands can be enabled/disabled from the UI script with a meaningful mnemonic.
- On form level, it is possible to specify if a Save command has to be performed prior to running the Form Command.
- It is possible to hide Form Commands on dialogs (not displayed as button).

Print sessions are also implemented as Form Commands. The bind type for print sessions is Print.

7.2 Implementing a form command

There are three steps to add a Form Command to your form:

1 In the forms session select the menu item Form Commands and insert a new Form Command:



The fields on this form have the following meanings:

Field	Purpose
ID	Just an internal number to differentiate between Form Commands.
Sort Sequence	Indicates the order for the buttons and menu items to appear.
Bind Type	For non-dynamic forms, this type can be set to Form or Field. When set to Field a special field of type Button should be created on the form and there is no button created on the right side of the form.
Activate a	Specifies whether a menu, session or function should be started.
Menu / Session / Function	Code of the Menu, Session or Function.
Shortcut	Shortcut key assigned to the command.
Label	Label that should be displayed on the button or menu item
Help Topic	Help topic for this Form command.
Separator	Option indicates whether a separator will be added after this button or menu item.
Show as Button	Checkbox that indicates whether the command is shown as a button on a single-occurrence form.
Authorization Group	The minimal authorization a user needs to perform this Form Command.
Execute Save	Indicates whether the standard program should perform an update (update.db) before this command is started, or not (Default is Yes).
Command Availability	An option that provides a means for automatically enabling and disabling commands according to the number of records selected (this is only meaningful in an overview session).

2 Dump the form.

3 Add a new function in your UI script:

```
function extern test.function()
{
        message("This is the test function")
}
```

This function should have the name as specified in the above session and should be extern. If these requirements are not fulfilled, an error message will appear:

7.3 Compatibility

The old form definitions are converted to the new definition with a conversion tool. After this conversion, the User Options, continu.process and form zooms are all converted to the new Form Commands.

7.3.1 User options

The User Options are converted to Form Commands of type *function*. The name of this function is exec.user.0 (or another number for other User Options). The standard program provides an extern function exec.user.0 that executes the before.-, on.- and after.choice of the choice.user.0 section.

7.3.2 Continue.process

The cont.process command is also converted to a Form Command of type function. The name of this function is exec.cont.process. The standard program provides an extern function exec.cont.process that carries out the before-, on- and after.choice of the choice.cont.process section.

7.3.3 Form zooms

The Form Zooms are also converted to a Form Commands, but they are of type session. The name of this session is taken from the zoom field of the choice field.

7.4 Disabling a form command

Disabling Form Commands is very simple: The argument to disable.commands is the name of the function, the code of the session, or the code of the menu. For example:

```
disable.commands(“test.function”, “ttadv3500m000”)
```

Note that Form Commands can also be enabled and disabled by the standard program according to the number of selected occurrences.

8 User interface checklist

8.1 Introduction

This checklist should be used as a validation tool for examining forms/windows in the BAAN V environment. The statements in the checklist are derived from the Standard UI and Microsoft UI guidelines.

8.2 Usage

Processing sessions and print sessions are to be considered as single-occurrence windows in this checklist.

8.3 Statements

8.3.1 Windows look

Sizes ,dimensions, and positioning

- Initial window size shows all essential information and action buttons.
- Initial window size is not bigger than necessary.
- Initial window size is big enough to show the toolbar and menu bar, if present, on one row each.
- Window objects (buttons, text fields, menu(item)s, check boxes and so on) which present user or system data are wide enough to show all possible occurrences.
- Resizing the window is prohibited (disabled) if it does not result in enlarging or narrowing the information view[1]. Specifically, dialog and single-occurrence windows may not be resized, but multi-occurrence windows may.
- Scrollbars are present if the information view exceeds the actual window view.
- All editable objects (combo boxes, text fields, check boxes, and so on) and clickable objects are horizontally and vertically aligned.

Toolbar, menu bar, status bar

- A dialog does not contain a toolbar, a menu bar or a status bar.

Labels and titles

- The window title briefly describes the type of information shown and/or its functional use and does not contain the words Maintain, Display or Edit.
- All visible text is written in regular font (no underlining, reverse video, italic, bold, and so on)

[1] information view: part of the virtual information space the user is authorized looking at.

Group boxes and checkboxes

- Information is functionally clustered and positioned within group boxes if the window does not hold a grid.
- The title of a group box describes a common feature of its items and does not contain the word Data.
- All group boxes within a window have the same width. If group boxes are positioned next to each other, the total width of them is equal to the width of the widest group box in the window.
- The maximum number of child-clusters[2] in a group box is seven.
- The minimum number of child-clusters in a group box is two.
- The group box labels and form page labels are unique within the window view and they are identical to similar and/or comparable labels used in other windows (views). These labels do not contain an index (1,2,3,...a,b,c…).
- All checkboxes are positioned within a group box.
- Checkboxes are not preceded by a label unless it is the first checkbox of a cluster of functionally related checkboxes, and the label describes a mutual or collective feature.
- Checkboxes are succeeded by a descriptive label. This label describes in a positive sense what option is chosen when the checkbox is marked.

Text fields and other editable fields

- Text fields and drop-down combo boxes are preceded by a descriptive label and may be succeeded by an additional label describing a currency, a unit, or a format.

Buttons

- The label on a text button describes the action it performs with a single verb (infinitive).
- A button may only have the label Cancel if it is capable of ending a dialog and an can undo of all actions performed so far in that dialog.
- A button may only have the label Close if its only action is to close the window.
- A window can only show one of the buttons Close, Continue or OK at the same time.
- A window does not show a Close and a Cancel button at the same time.
- The label on a text button or menu entry contains an ellipsis (...) if using the option starts a dialog in which more input from the user is required.
- An OK button or a button to perform a specific action such as printing on a print dialog) is the top (first) button in a window.
- A help button is the bottom (last) button in a window.

[2] Child-cluster: within a cluster/group box/frame, a label and its text field count as one child-cluster.

Multi-occurrence specific

- A multi-occurrence window does not contain text buttons.
- In a multi-occurrence window, each column is preceded by a descriptive label.
- A multi-occurrence window does not contain editable fields unless its replaces single-occurrence window.
- A multi-occurrence window with editable fields holds no more than three editable objects and one display object per row. The maximum width is 80 characters.
- A multi-occurrence window with editable can not have more than one form page.

System messages and help text

- Error messages describe the erroneous situation with a short, unambiguous text.
- A help option is available when presenting an error message. The help text describes the cause of the error and explains how the situation can be avoided or resolved.

8.3.2 Windows feel

- A session that starts from the menu browser, shows a multi-occurrence window.
- A multi-occurrence window initially presents all data in the information view unless the developer can expect more than 1000 entries with the same primary index key. In this case, the developer should create the possibility to narrow the view using a non-editable text field (view field) showing a particular key value which all presented entries share.
- The Insert and Modify options on a multi-occurrence window start a single-occurrence session.
- A zoom button on a single-occurrence form will show a multi-occurrence window that displays only those occurrences that may be selected.
- The default action (double mouse click) on a multi-occurrence entry is Modify if the window is not started using a zoom/browse button.
- The default action (double mouse click) in a multi-occurrence window started by using a zoom/browse button is closing the multi-occurrence window and returning the selected entry to its parent.
- A single-occurrence window is closed if its multi-occurrence parent is closed.
- A warning is given to if the user intends to close a window holding edited data.
- Window objects (buttons, text fields, menu(item)s, checkboxes, and so on) that can not be modified must be made read-only.
- Window object (buttons, text fields, menu(item)s, checkboxes, and so on) that are (temporarily) not applicable or irrelevant due to a combination of values in the form must be disabled (grayed out).